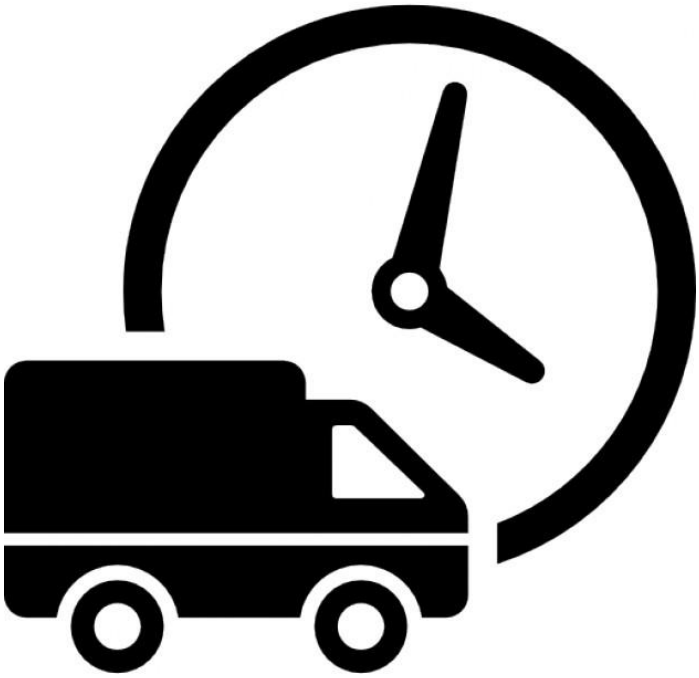
REORGANITZACIÓ DE LA LOGÍSTICA DE DISTRIBUCIÓ D'UNA EMPRESA

Memòria final



Autor: Arnau Soler Facundo
Data: 17/01/2016
Director: Josep Gubianas
Ponent: Albert Oliveras Llunell

Resum

Català

Cada vegada són més les empreses de la societat actual que comencen a créixer i necessiten replantejar la seva gestió de recursos. Un programa molt conegut i molt utilitzat per gestionar els recursos d'aquestes petites i mitjanes empreses és el *SAP Business One*.

El *SAP Business One* és un programa de gestió empresarial que engloba en un sol entorn tots els mòduls necessaris per a la gestió d'una empresa com per exemple, finances, recursos humans, compres, vendes, etc.

Amb l'ajuda d'aquest programa, la petita i mitjana empresa pot créixer de forma més ràpida i eficient, però cal tenir en compte, que hi ha aspectes tant importants com la logística de distribució que no és capaç de gestionar. La solució passa per generar un programa compatible amb *SAP Business One*, que permeti gestionar la logística de transport.

El problema d'aquestes solucions és que degut a l'alt nivell de competència del mercat són totes privades i poden arribar a tenir costs molt elevats.

Per tot això, crec que el programa de reorganització de la logística de distribució de l'empresa que generem al llarg d'aquest projecte, pot ser una eina de gran ajuda per a tots aquests empresaris que comencen a créixer.

Español

Cada vez son más las empresas de la sociedad actual que empiezan a crecer y necesitan replantear su gestión de recursos. Un programa muy conocido y utilizado para gestionar los recursos de las pequeñas y medianas empresas es el *SAP Business One*.

El *SAP Business One* es un programa de gestión empresarial que engloba en un solo entorno todos los módulos necesarios para la gestión de una empresa como por ejemplo, finanzas, recursos humanos, compras, ventas, etc.

Con la ayuda de ese programa, la pequeña y mediana empresa puede crecer de forma más rápida y eficiente, pero se debe tener en cuenta, que hay aspectos tan importantes como la logística de distribución que no es capaz de gestionar. La solución consiste en generar un programa compatible con *SAP Business One*, que permita gestionar la logística de transporte.

El problema de esas soluciones es que debido al alto nivel de competencia que hay en el mercado, son todas privadas y pueden llegar a tener un coste muy elevado.

Por todo esto, creo que el programa de reorganización de la logística de distribución de una empresa que generamos al largo de este proyecto, puede ser una herramienta de gran ayuda para todos aquellos empresarios que empiezan a crecer.

English

More and more companies in the current society are starting to grow and need to redesign their management of resources. A really well known program used to manage the resources of small and medium-sized enterprises is *SAP Business One*.

SAP Business One is an enterprise resource planning program that brings together in a single environment all the modules necessary for the management of a company, such as, finance, human resources, purchasing, sales, etc.

With the help of this program, small and medium-sized enterprises can grow quickly and efficiently, but we have to take into account, that there are aspects as important as distribution logistics, that it can't manage. The solution is to generate a program that is compatible with *SAP Business One*, which allows to properly manage transport logistics.

The problem with this kind of solutions is that, due to the high level of competition in the market, they are all private and can be very expensive.

That's why, I believe that the program of distribution logistics that will be generated along this project, can be a great tool to help all those entrepreneurs who begin to grow.

Índex de continguts

Índex de taules

Índex d'il·lustracions

1 Introducció

El problema del viatjant de comerç[1] és un problema molt comú en l'àmbit de la ciència de la computació. Va ser formulat per primera vegada al 1930 i ha estat un dels problemes d'optimització més estudiats. Es basa en trobar la ruta més curta entre un conjunt de ciutats passant per totes elles i finalitzant a la ciutat d'inici. Actualment té diverses aplicacions com per exemple el disseny d'una línia de metro o la recollida de comandes d'un magatzem (1) (2).

A la societat actual la majoria d'empreses que comencen a créixer es troben davant d'una problemàtica molt comuna, dissenyar i gestionar la seva logística de transport. Planificar les rutes de transport d'una empresa pot ser una tasca molt complexa ja que hi ha molts paràmetres a tenir en compte. Si la planificació no és molt bona, fer-ho manualment és perillós, ja que el dia que s'hagin de realitzar canvis s'haurà de refer gran part del disseny. Al mercat existeixen múltiples solucions per aquest tipus de problemes, però com que no totes les empreses es gestionen els recursos de la mateixa forma, no existeix una solució global.

Quant a la gestió de recursos, un gran nombre de petites i mitjanes empreses, que estan en creixement, utilitzen un programa de gestió de recursos empresarials molt conegut, el *SAP Business One* (o SBO). És un programa molt complex, que engloba un conjunt de funcionalitats indispensables per a l'empresa en un sol entorn com per exemple, finances, gestió de clients, inventaris, recursos humans... Tot i ser capaç de solucionar la majoria d'aspectes relacionats amb la gestió d'una empresa, per defecte no porta incorporat cap mòdul capaç de tractar la logística de transport (3) (4).

La logística de transport és un tema amb el que s'hi guanyen la vida moltes empreses, que es dediquen a dissenyar programes per optimitzar-la i automatitzar-la. Una solució interessant és la de *Routing Reparto* (5), amb un conjunt d'excels permet planificar la teva ruta de transport ideal, agafant

[1] En anglès TSP o *Travelling salesman problem*

dades de localització a temps real de *Google* i permetent vàries configuracions. A més a més s'adapta a diferents tipus d'empresa i producte.

La pàgina web anterior és una de les moltes solucions que existeixen, però aquest projecte va enfocat a una empresa que utilitza SBO com a sistema de gestió de recursos, així que és important que la solució que es faci servir sigui compatible amb aquest. Una empresa interessant és *SGI Solutions* (6), que desenvolupa diferents programes (o *Add-ons*), que s'instal·len a SBO i hi afegeixen un conjunt de funcionalitats com per exemple, la gestió de magatzems, projectes o transport.

El problema d'aquestes solucions és que degut a l'alt nivell de competència que hi ha al mercat són totes privades. Es poden arribar a pagar molts diners per al seu desenvolupament.

1.1 Formulació del problema

Al trobar-se dins un mercat tan competent, moltes empreses opten per generar la seva pròpia solució. Però al no disposar dels especialistes o del temps suficient, moltes vegades els programes acaben sent poc eficients ja que, amb el sol fet de que es compleixi el funcionament mínim ja en tenen prou.

En aquest projecte treballem amb Seidor, una empresa que és *partner* de SAP AG, és a dir, que té la llicència per vendre els seus programes i que es dedica a fer-ne la instal·lació i manteniment als seus clients. Més concretament, ens centrarem en *SAP Business One* (o SBO) que és una de les solucions que ofereix SAP AG a les petites i mitjanes empreses per gestionar els seus recursos.

Alguns clients de l'empresa requereixen un sistema de gestió per la seva logística de transport, així que s'ha dissenyat un *Add-on* que s'encarrega de dur-ho a terme. El problema és que cada vegada s'hi han afegit més condicions i al no haver-ho plantejat bé des d'un principi fa que sigui difícil realitzar-hi qualsevol modificació. Tot i així el problema principal no és aquest, sinó que no s'ha tingut en compte l'eficiència, i a mesura que les empreses

client augmenten el seu nombre de clients o han de modificar alguna ruta fa que sigui molt complicat.

Actualment hi ha un grup de persones que s'encarreguen d'agafar el conjunt de comandes a servir i de forma manual dissenyen les rutes. El que es vol aconseguir amb aquest projecte és dissenyar un sistema capaç de crear i gestionar les rutes automàticament. Addicionalment, es vetllarà perquè els resultats que obtingui la solució siguin molt propers a la solució òptima, cosa que actualment no es tenia en consideració degut a la gestió manual.

Per tal de modificar el mínim el programa (o *Add-On*) que ja hi ha creat, es generarà un programa extern que connectarà amb SBO, hi farà les consultes pertinents i retornarà una solució. En aquest cas ens interessa que calculi les rutes de transport i les guardi en un fitxer de text. L'*Add-On* seguirà realitzant les seves tasques però ara disposarà d'un nou conjunt de rutes calculades amb les que treballar.

Les rutes que volem construir son les de servei, on s'han de servir un conjunt de productes a un conjunt de clients, és a dir, s'han de repartir les comandes. El programa s'executarà a diari o en la freqüència que convingui i calcularà i desarà les noves rutes a SBO.

Per aconseguir una generació automàtica de rutes de transport, s'utilitzarà una metaheurística que veurem definida més endavant. Davant la possibilitat que el nombre de clients i productes sigui molt alt, els mètodes complets basats, per exemple, en programació lineal, han estat descartats degut a la seva poca escalabilitat.

1.2 Contextualització

En aquest projecte tractem una variació del problema del viatjant de comerç. Hi ha moltes solucions per aquest problema al mercat. Aquestes aplicacions fan servir tècniques d'optimització combinatòria per trobar la millor solució. Més endavant ho veurem amb més detall.

1.3 Actors implicats

Els actors implicats al projecte són aquelles persones o organitzacions a les quals els hi pot interessar.

1.3.1 Dissenyador, Desenvolupador i Tester

Les tasques de dissenyador, desenvolupador i tester les realitzaré jo mateix, ja que sóc l'encarregat de dur a terme el projecte.

1.3.2 Director i tutor del projecte

El director i tutor d'aquest projecte és en Josep Gubianas, la seva missió és la de supervisar que el projecte compleixi amb el calendari establert i que s'assoleixin els objectius marcats. Addicionalment, pot ajudar i guiar al desenvolupador en cas que sigui necessari.

1.3.3 Ponent del projecte

El ponent del projecte és l'Albert Oliveras, la seva missió és la d'ajudar al dissenyador, desenvolupador i tester durant el transcurs del projecte. L'ajuda pot ser de vàries formes com per exemple, quan la programació queda estancada li proporciona alguna pista sobre com avançar, li recomana algun document acadèmic per llegir o altres.

1.3.4 Seidor

L'empresa per la qual es dissenya el projecte, que és *partner* de SAP AG, pot estar interessada en la solució obtinguda amb aquest projecte. Disposarà d'un programa extern compatible amb SBO i amb el seu *Add-On* de gestió de transport, que permetrà generar un conjunt de rutes de servei de comandes per als seus clients complint certes restriccions de forma automàtica.

1.3.5 Empreses client de Seidor

Els clients de Seidor que fins ara feien servir la solució de transport actual, disposaran d'un programa extern que els hi generarà de forma automàtica i eficient un conjunt de rutes de servei, que a la llarga els hi podria suposar un estalvi de temps i diners.

2 Estat de l'art

2.1 El problema del viatjant de comerç

El problema del viatjant de comerç[2] és un dels problemes d'optimització combinatòria més estudiats. Va ser formulat matemàticament al segle 18 pel matemàtic irlandès Sir William Rowam Hamilton i el matemàtic britànic Thomas Penyngton Kirkman, però la seva forma general no va ser estudiada fins al 1930 a Viena i a Harvard per Karl Menger un matemàtic de nacionalitat austríaca i americana.

Consisteix en un comerciant que surt d'una ciutat i ha de visitar un conjunt de ciutats exactament una vegada i retornar a la ciutat d'origen. L'objectiu és trobar el recorregut de mínima distància o cost. Donat que el comerciant ha de visitar n ciutats, el nombre total de rutes possibles per visitar-les totes és de $(n-1)!$.

Es genera un graf on cadascun dels nodes representa una ciutat i les arestes són les connexions entre elles. Una solució vàlida per resoldre el problema del viatjant de comerç és la que correspon a un cicle hamiltonià del graf. Si ho apliquem al nostre cas, per a obtenir una ruta de transport, els nodes serien els clients i les arestes les connexions entre ells.

Es pot definir com un problema simètric on les arestes que connecten cada un dels nodes tenen el mateix cost, o com un problema asimètric on les arestes són diferents en cada connexió de tal manera que passem a tenir un graf dirigit.

La seva versió decisional pertany a la classe de problemes NP-complets, que representen el subconjunt de problemes de NP més difícils de resoldre. NP és la classe de complexitat formada pel conjunt de problemes de decisió, que es poden resoldre en temps polinòmic i de forma no determinista per una màquina de Turing. (1) (2).

[2] En anglès TSP o *Travelling salesman problem*

2.2 Problema de rutes de vehicles

El problema de rutes de vehicles[3] és una generalització del problema anterior, així que també forma part de la família de problemes d'optimització combinatòria. Va ser formulat per George Dantzig i John Ramser al 1959 per solucionar un problema amb les entregues de gasolina. Al 1964 va aparèixer una millora de l'aproximació de Dantzig i Ramser a mans de Clarke i Wright coneguda com l'algorisme d'estalvis, el veurem en més detall al següent apartat.

Consisteix en dissenyar el sistema d'entregues d'una companyia, donats un conjunt de vehicles i un conjunt de clients a servir. Aquest conjunt de clients està interconnectat mitjançant un conjunt de carreteres. Una o més de les ciutats són els magatzems des d'on sortirà el producte a distribuir. L'objectiu és obtenir un conjunt de rutes, una per cada vehicle, tal que es serveixi a cadascun dels clients amb el mínim cost, ja sigui en temps, diners, distància o una combinació de tots (7) (8).

2.2.1 Variants del problema de rutes de vehicles

Hi ha vàries variants del problema de rutes de vehicles que afegeixen certes restriccions. Una d'elles és coneguda com a problema de rutes de vehicles amb capacitats[4] on cadascun dels vehicles que s'encarrega de la distribució té una capacitat màxima.

Una altra variant, és el problema de rutes de vehicles amb finestres de temps[5] on cadascun dels clients a servir té un horari d'obertura i un horari de tancament, és a dir, el servei del producte s'ha de fer durant aquest interval.

La fusió d'aquestes dues variants és la que ens interessa resoldre al nostre projecte ja que tindrem una flota de vehicles amb una capacitat màxima i un conjunt de clients amb uns horaris concrets (8)

[3] *En anglès VRP o Vehicle Routing Problem*
[4] *En anglès CVRP o Capacitated Vehicle Routing Problem*
[5] *En anglès VRPTW o Vehicle Routing Problem with Time Windows*

2.3 Heurístiques

Les heurístiques són tècniques dissenyades per resoldre un problema quan els mètodes clàssics, com per exemple els algorismes exactes, són massa lents o fallen alhora de trobar una solució exacta, en aquest últim cas ens ajuden a trobar una solució aproximada. Es poden classificar en heurístiques clàssiques i metehaurístiques. En aquest apartat veurem en detall un tipus d'heurística clàssica, la constructiva, i les metaheurístiques.

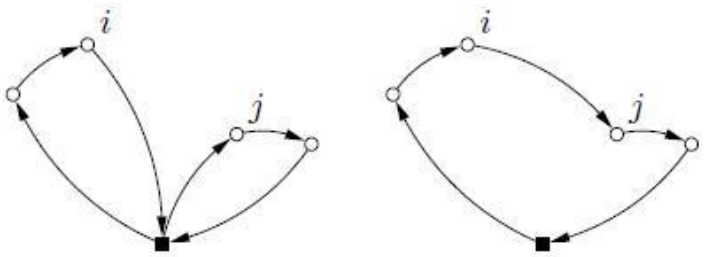
2.3.1 Heurístiques constructives

Al 2002 Laporte i Semet van definir aquest tipus d'heurístiques com: "*Una heurística constructiva construeix gradualment una solució factible alhora que manté controlat el cost de la solució, però no obté una fase de millora de per si*". Les heurístiques constructives per resoldre problemes de vehicles de rutes es divideixen en tres classes: d'estalvi i d'inserció i de cerca local (9).

2.3.1.1 Heurística d'estalvi

L'heurística d'estalvi va ser proposada primerament al 1964 per Clarke i Wright i és una de les més conegudes. Es fa servir a problemes on el nombre de vehicles no és fix i funciona en grafs dirigits i no dirigits.

Si en una solució dues rutes diferents $(0,\ldots,i,0)$ i $(0,j,\ldots,0)$, on 0 representa el magatzem, poden ser combinades formant una nova ruta $(0,\ldots,i,j,\ldots,0)$ com podem veure a la següents il·lustració:



Il·lustració 1: Algorisme d'Estalvis

L'estalvi en distància que obtenim d'aquesta unió és:

$$S_{ij} = C_{i0} + C_{oj} - C_{ij}$$

Que representa l'estalvi que suposa servir els clients (i, j) en una ruta sola a diferència de servir-los individualment des del magatzem.

Aquest algorisme construeix les rutes que formaran la solució de forma seqüencial, realitzant unions entre els clients tal que resultin amb un millor estalvi, sempre respectant les restriccions del problema. (10)

2.3.1.2 Heurística d'inserció

Les heurístiques d'inserció construeixen les solucions inserint un client a cada iteració, poden construir una ruta a la vegada si treballen seqüencialment o vàries en cas de treballar paral·lelament. La forma d'escollir el client a col·locar i on col·locar-lo és el que diferencia les heurístiques d'inserció.

L'heurística que veurem i que farem servir per obtenir una solució inicial al nostre projecte és la *Push Forward Insertion Heuristic* (o PFIH). Inicialment tenim un conjunt de clients a col·locar i una llista de rutes buida. Per representar una ruta fem: $(i_0, i_1, i_2, \dots i_m)$ on $i_0 = i_m = 0$ (on 0 representa el magatzem). Així doncs la primera ruta de la llista ja s'inicialitza de la següent forma: (i_0, i_m), només amb el magatzem.

L'objectiu de l'algorisme és inserir els clients del conjunt a una ruta fins que no es pugui, degut a l'incompliment d'una o més restriccions, com per exemple, la capacitat dels vehicles. Un cop tenim una ruta plena, la guardem i en generem una de nova, així fins a haver col·locat tots els clients.

Per un client u pendent d'afegir, la fórmula que s'utilitza per calcular la millor posició d'inserció a la ruta és la següent:

$$C1(i(u), u, j(u)) = min[C1(i_{p-1}, u, i_p)], p = 1, \dots, m$$

La inserció d'un client u entre les posicions $(i(u), j(u))$ de la ruta en qüestió serà la que impliqui un menor cost min que veurem més endavant com calcular.

Veiem que la inserció és entre i_{p-1} i i_p, la p representa el conjunt de tots els clients en ruta, inclòs el magatzem, que tenim a l'inici i al final. Si tenim una ruta $(0,1,0)$ i un client 2, el podem inserir i crear les següents rutes $(0,2,1,0)$ i $(0,1,2,0)$. Les posicions serien entre (i_0, i_1) i (i_1, i_0) respectivament. D'aquestes dues insercions escolliríem la millor segons el resultat de min.

Cal tenir en compte que una inserció de u entre i_{p-1} i i_p pot alterar potencialment els temps d'arribada a la resta de clients $(i_p, \dots i_m)$, és a dir, quan inserim un client entre dues posicions, els temps d'arribada a la resta s'han de tornar a calcular per tal d'assegurar que es segueixin respectant les restriccions.

Un cop provats tots els possibles clients a inserir, és a dir que compleixin les restriccions, en un ruta, seleccionarem la inserció més òptima mitjançant la següent fórmula:

$$C2(i(u*), u*, j(u*)) = optimum[C1(i(u), u, j(u))], u\ viable\ i\ pendent\ d'afegir$$

$C2$ ens retorna la inserció més òptima, en quant a cost, de les obtingudes amb la fórmula anterior, $C1$. Acte seguit veurem que significa una inserció $optimum$.

Col·loquem el client que retorna $C2$ a la ruta a les posicions indicades i tornem a repetir el procés. Un cop no es puguin inserir més clients a una ruta, la tanquem i n'obrim una de nova. L'algorisme acaba quan no queda cap client per col·locar.

Veiem ara més en detall quin és el càlcul que utilitzem per les fórmules $C1$ i $C2$. Per calcular $C1$ es fa servir la fórmula:

$$C1(i,u,j) = \alpha_1 C11(i,u,j) + \alpha_2 C12(i,u,j),\ \alpha_1 + \alpha_2 = 1$$

$$\alpha_1 \geq 0, \alpha_2 \geq 0$$

Aquesta fórmula està formada per dues altres fórmules que són les següents:

$$C11(i,u,j) = d_{iu} + d_{uj} + \mu d_{ij}\ , \mu \geq 0$$

$$C12(i,u,j) = b_{j_u} - b_j$$

Per tal d'entendre millor el seu funcionament, veiem que significa cadascuna de les variables. Primerament tenim d_{iu}, que representa la distància entre els clients (i,u), la resta de d representen el mateix.

b_j és el moment que el camió arriba al client j, és a dir, quan un camió surt cap al client j, surt a una hora concreta, tarda un temps a fer el servei, en cas que estigui a un altre client i té un temps de viatge, tot això és el que determina l'hora d'arribada al client j. b_{j_u} és el nou temps d'arribada al client j, després de la inserció de u davant de j.

Per calcular $C2$ es fa servir la fórmula:

$$C2(i,u,j) = \Lambda d_{0u} - c1(i,u,j), \Lambda \geq 0$$

Disposem de 3 variables, α_1, α_2 i μ que són paràmetres configurables de l'algorisme que ens permeten modificar el càlcul segons ens interessi. Per exemple, si escollim com a paràmetres $\mu = \alpha_1 = \Lambda = 1$ i $\alpha_2 = 0$, la segona fórmula, $C2(i,u,j)$ es converteix en l'algorisme d'estalvis que hem vist anteriorment. $C2(i,u,j)$ representa l'estalvi en distància de servir el client u a la mateixa ruta que els clients (i,j) a diferència de servir-lo directament des del magatzem.

Aquest tipus d'heurística d'inserció intenta maximitzar el benefici derivat de servir un client en una ruta en construcció en comptes de una ruta directa, és a dir, haver de servir-lo des del magatzem.

Així doncs amb els paràmetres α podem modificar la primera fórmula, $C1$, per tal de que es basi només amb la distància, només amb el temps o en ambdues variables. I amb els paràmetres μ i Λ podem alterar la forma amb que l'algorisme selecciona els clients.

Finalment, un cop sabem com es calcula $C1$ i com es calcula $C2$, només queda comentar que min seleccionava el menor resultat de $C1$, i $optimum$ el major resultat de $C2$.

El que intenta aquesta heurística és obtenir la posició d'inserció d'un client tal que es minimitzi la distància i el temps que es requereixen al servir-lo. (11)

2.3.1.3 Heurística de cerca local

Les heurístiques de cerca local són les que parteixen d'una solució inicial, i la modifiquen realitzant un conjunt d'operacions seqüencials per tal d'intentar produir una millor solució. La solució inicial es pot calcular per exemple, fent servir una heurística d'inserció com la que hem vist a l'apartat anterior.

Sempre es manté la millor solució obtinguda fins al moment. Es van fent canvis a aquesta solució i, en cas de trobar-ne una de millor, es substitueix per l'antiga i es torna a iniciar el procés.

Aquest tipus d'heurístiques també són conegudes com a heurístiques de millora ja que, el seu funcionament es basa en realitzar un seguit d'operacions amb l'únic objectiu de millorar la solució.

L'heurística que veurem i que també farem servir al nostre projecte s'anomena $\Lambda - interchange$. Es basa en l'intercanvi de clients entre diferents rutes de vehicles d'un veïnat. Un veïnat o espai de solucions, consisteix en el conjunt total de les rutes que formen la solució.

El nombre de clients a intercanviar entre cadascuna de les rutes el defineix Λ. Per exemple, si $\Lambda = 1$, podem fer els intercanvis de clients $(1,0), (0,1)$ i $(1,1)$. L'operador $(1,1)$ en un parell de rutes $(R1, R2)$ indica que movem un client de $R1$ a $R2$ i un client de $R2$ a $R1$. La resta d'operadors es defineixen de la mateixa forma. Perquè un intercanvi sigui vàlid ha de resultar amb una millora a la solució general.

Partint d'una solució inicial, apliquem tots els intercanvis possibles i seleccionem la següent millor solució. Hi ha dues estratègies de selecció, *First Best* (FB) o *Global Best* (GB), la primera selecciona com a millor solució la primera que redueixi el cost respecte l'actual, la segona en canvi, explora tot l'espai de solucions i selecciona la millor de totes.

L'algorisme finalitza quan s'explora tot l'espai de solucions i no es pot obtenir una millora en la solució (9) (12) (13).

2.3.2 Metaheurístiques

Com hem vist anteriorment les heurístiques són tècniques dependents d'un problema, el seu objectiu és obtenir el màxim rendiment en aquest, però degut a que a vegades són massa avaricioses es queden encallades localment i per tant, fallen alhora d'obtenir resultats propers a la solució òptima general.

Aquí és on entren en joc les metaheurístiques, una àrea de recerca molt popular durant els últims 20 anys. Aquestes ja no són dependents del problema de tal forma que, a vegades, fins i tot accepten una pitjor solució per tal de poder explorar completament l'espai de solucions, tal com veurem al següent apartat amb l'algorisme *Tabu Search*.

Aquesta capacitat de no només buscar un solució millor, és la que permet que l'algorisme explori tot l'espai de solucions i així es pugui obtenir una aproximació a l'òptim general de la solució (14).

2.3.2.1 Tabu Search

Estratègia de cerca basada en memòria. Es divideix en dues parts, intensificació i diversificació. Partim d'una solució inicial obtinguda amb qualsevol heurística, com per exemple, una d'inserció, com el PFIH que hem vist anteriorment.

Un cop obtinguda la solució inicial comencem el procés d'intensificació. Aquest consisteix en aplicar una heurística de cerca local a la solució obtinguda, com per exemple, la tècnica vista anteriorment, el $\Lambda - interchange$.

Quan ja no hi ha forma d'obtenir una millor solució, és a dir, ens hem quedat estancats localment en un espai de solucions, comença el procés de diversificació. Es porten a terme una sèrie de moviments aleatoris anomenats $hops$ per tal de moure'ns cap a un altre espai de solucions. Més endavant ho veurem més en detall.

El procés es va repetint fins que es compleix una condició d'aturada concreta, com per exemple, un màxim nombre d'iteracions o un nombre concret d'iteracions sense obtenir beneficis.

L'algorisme està basat en dos tipus de memòria, d'emmagatzematge i de freqüència, per tant, fa servir vàries estructures de dades. La primera i la més important és la $llista\ tabú$, és una cua de mida fixa que emmagatzema moviments i solucions trobades al llarg de l'execució, s'utilitza per tal d'evitar cicles durant aquesta.

Per emmagatzemar un moviment ho fa de la següent forma:

$$< client1, R1, posició1, R2, posició2 >$$

On $R1$ i $R2$ són les rutes que fan l'intercanvi, $client1$ és el node que intercanviem i les posicions són les que aquest ocupava i ocupa a les rutes respectives.

Una altra informació que es guarda a la $llista\ tabú$ són les solucions obtingudes al llarg del procés, per tal de fer-ho de forma simple ho fem de la següent forma:

$$Ruta\ 1\text{: } 0,7,8,9,10,34,5,0,390$$
$$Ruta\ 2\text{: } 0,6,12,20,45,17,0,325$$

Es guarden per cadascuna de les rutes els clients que en formen part, es delimita per zeros que representen el magatzem. Al final es guarda el cost de la ruta, veurem perquè més endavant.

De tots els moviments i solucions que emmagatzemem portem el control del nombre de vegades que els hem emmagatzemat, és a dir, tenim en compte la freqüència en que fem un moviment o obtenim una solució.

Una altra estructura important és la $llista\ de\ candidats$, aquesta conté el conjunt de solucions que hem anat trobant que es podran fer servir en una futura intensificació, és a dir, solucions que han suposat una millora durant l'execució. Podem reaprofitar l'estructura anterior utilitzada per guardar les solucions tabú. Aquí és on ens interessa tenir el cost de cadascuna per tal de poder començar la pròxima ronda d'intensificació amb la millor de la llista.

Un cop definides les estructures veiem més en detall el funcionament dels processos d'intensificació i diversificació. Durant la intensificació agafem una solució inicial i apliquem el mètode $\Lambda - interchange$, qualsevol solució que resulti amb un GB es guarda a la $llista\ de\ candidats$ per futures intensificacions.

Cada moviment o nova solució obtinguda es compara amb la llista tabú per veure si està prohibit o no, i així evitar cicles. Hi ha casos on es permet portar a terme una acció tabú, només si es compleix amb un cert criteri com per exemple que el moviment prohibit resulti amb un GB. Al fer servir el mètode GB per $\Lambda - interchange$, qualsevol solució que obtinguem sempre complirà amb el criteri i per tant, no es veurà afectada per la $llista\ tabú$, però si que augmentarà la freqüència de visites d'aquesta.

Cada moviment o solució que visitem, que estigui a la *llista tabú*, fa que augmenti la freqüència de visita per aquest element. Si es visita masses vegades un mateix element, finalitza la intensificació ja que, estem donant voltes sobre un mateix espai de solucions de forma innecessària, és a dir, estem estancats localment.

Un cop s'ha cercat tot el veïnat, s'agafa la solució que ha quedat estancada i es comença el procés de diversificació. Aquí es fan un seguit de moviments aleatoris i es van guardant les solucions, sense comprovar que siguin millors, simplement interessa explorar nous espais de solucions. Totes les solucions obtingudes es guarden a la *llista de candidats* i s'ordenen per cost. Si la solució està a la *llista tabú* no es guarda a la *llista de candidats*.

Acabat el procés de diversificació es selecciona el millor candidat de la llista, que no sigui tabú, i es torna a començar una nova ronda d'intensificació. El procés es va repetint fins que es compleix alguna de les condicions d'aturada esmentades anteriorment.

Amb aquest algorisme aconseguim desplaçar la nostra solució fora de la cerca local en un sol espai de solucions i així explorar altres zones per tal d'obtenir una aproximació a l'òptim general. Per a fer-ho hem vist que s'executa un procés que considera solucions pitjors que l'actual per tal d'obtenir una millora en un futur (12) (13).

3 Definició de l'Abast

3.1 Abast

Per tal de trobar un conjunt de solucions que retornin un bon resultat, cal tenir ben controlats els passos a seguir abans de començar a elaborar el codi. És una tasca molt important ja que és la que marcarà la pauta a seguir al llarg del projecte.

Primerament s'han de definir molt bé les dades d'entrada. El conjunt de dades més importants a emmagatzemar són les geolocalitzacions de cadascun dels clients a servir, ja que són les que ens permetran situar-los al mapa. Obtindrem aquesta informació a través de l'eina *Google Maps*. La informació obtinguda serà gravada a les fitxes de client de SBO, d'aquesta manera disposarem de la informació de geolocalització juntament amb la resta de dades necessàries en un sol lloc.

Acte seguit s'han de definir les restriccions que volem que es compleixin a les rutes de servei que obtindrà el programa. La primera és tenir en compte que cadascuna de les rutes serà servida per un únic vehicle i que aquest tindrà una capacitat màxima la qual no es podrà superar. La segona i última es basa en complir amb els horaris de servei que cada client té estipulats, és a dir, servir-li el producte durant la franja horària que ens indica. Disposem de tota la informació a SBO, a les fitxes de clients i vehicles.

L'últim pas consisteix en indicar els resultats que esperem obtenir. Volem que la solució tingui el mínim nombre de rutes, és a dir, que s'utilitzin el mínim nombre de camions, i volem que el cost del viatge de tots els vehicles, en quant a temps i distància, sigui el menor possible. Per tal de d'interpretar-ho de forma adequada farem servir un ordre lexicogràfic sobre el parell $(\#\, rutes, temps\ total)$ on $(2{,}120) < (4{,}90) < (5{,}90)$.

Un cop s'hagin obtingut les dades, s'hagin definit un conjunt de restriccions i es tingui clara la solució a obtenir, es farà un anàlisi per tal de trobar la millor forma de resoldre el problema. En aquest tipus de problemes ens podem trobar amb dues situacions, si el problema té una mida raonable es pot

solucionar amb mètodes complets, si la mida creix i el nombre de restriccions és elevat, llavors s'ha de solucionar amb mètodes incomplets.

La gestió de totes aquestes dades i restriccions es farà a través del programa extern que dissenyarem, que agafarà tota la informació necessària de SBO i generarà una solució de manera eficient. El programa generarà un seguit de consultes a SBO per obtenir informació dels vehicles que hi ha disponibles, el conjunt de clients i les comandes que se'ls hi han de servir. Amb aquesta informació farà les computacions necessàries per a obtenir el conjunt de rutes de servei que compleixi amb les restriccions introduïdes i les desarà a un fitxer de text.

3.2 Possibles obstacles

En aquesta secció es detallaran els possibles obstacles amb els que ens podem trobar al llarg del projecte i s'indicaran algunes solucions per tal de reduir-ne els efectes.

3.2.1 Dades obtingudes de *Google*

Tot i que *Google Maps* té enregistrades la majoria de localitzacions, podria ser possible que algun dels nostres clients no aparegui al mapa, o que tingui una localització inaccessible per un vehicle. Si es dóna el cas, s'haurà de tenir en compte i registrar una localització al nostre programa que s'aproximi a la del client en qüestió.

3.2.2 Compatibilitat amb el software de gestió empresarial

El programa de gestió *SAP Business One* que utilitza l'empresa és un producte al qual ens hem d'adaptar, ja que no tenim accés al seu codi. Això implica que si sorgeix un problema no tenim gaires opcions. La única possibilitat, és fer un bon anàlisi abans de començar a crear el nostre programa per tal de que, a la que ens posem a programar, no ens trobem amb cap impediment.

3.2.3 Errors de programació

En un projecte de software on el pes de la part computacional és tan elevat, poden sorgir molts errors de programació que farien que no obtinguéssim els resultats esperats. Per tal d'evitar aquesta situació, es mantindrà un registre

d'errors de manera que es pugui saber en tot moment que és el que no funciona i que és el que no havia funcionat anteriorment. Addicionalment, es realitzaran un seguit de proves a cada nova versió que es generi per assegurar que l'evolució del projecte sigui l'esperada.

3.2.4 Data límit d'entrega

Cal tenir en compte que el projecte s'ha de realitzar durant el semestre al qual s'ha matriculat, així que s'ha de controlar bé el temps. Per solucionar aquest problema es fixaran un seguit de fites a complir cada una o dues setmanes que seran validades conjuntament amb el director del projecte i també amb el ponent en alguns casos.

3.3 Metodologia i rigor

Anteriorment, hem vist que un dels possibles obstacles és que tenim una data límit per entregar el projecte, així doncs, serà necessari utilitzar una metodologia àgil per tal de minimitzar el risc al desenvolupar el programa.

La majoria de processos d'aquest estil estan orientats per a un grup de persones, on cadascú té la seva feina. En aquest projecte només hi ha tres participants, l'alumne, el director i el ponent, però podem utilitzar alguns conceptes dels processos àgils per realitzar les nostres tasques.

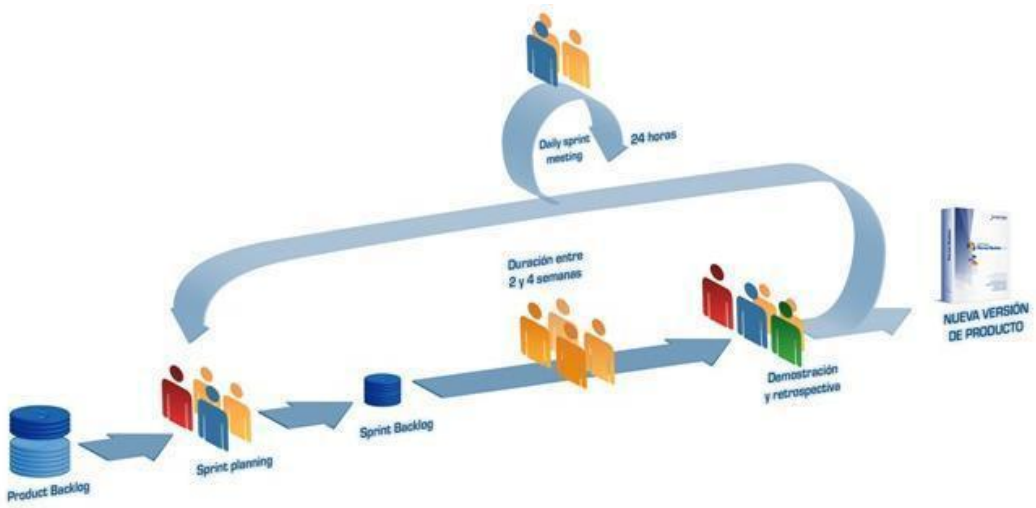
Primerament, destacar que el projecte no va dirigit directament a un conjunt de clients, sinó que és un programa que es genera per tal de millorar les solucions que aquests tenen actualment, així doncs, el client no serà el primer en fer la valoració del funcionament del programa, sinó que seran el director i el ponent del projecte.

Es portaran a terme una sèrie de reunions periòdiques per validar la bona evolució del projecte, tal com si fos una reunió amb el client. L'objectiu d'aquestes reunions, apart de validar l'evolució del projecte, és mantenir un feedback constant amb el director i el ponent, de forma que es puguin discutir els progressos des de diferents punts de vista.

Com que es disposa de poc temps per a realitzar el projecte, caldrà establir un conjunt de fites a assolir setmanalment. Emprant aquesta tècnica s'estableix un molt bon control de l'evolució del projecte, ja que cada setmana es revisa si s'han complert els objectius. En cas que no es compleixin, som a temps de rectificar ràpidament per tornar a l'estat que s'havia planificat des d'un principi.

Finalment, s'ha comentat que un dels obstacles més problemàtics amb els quals ens podem trobar són els errors de programació. Per evitar-los o reduir-los el màxim possible, cal utilitzar un procés de desenvolupament basat en proves. Si seguim aquesta metodologia aconseguirem reduir el nombre d'errors i la correcció del codi serà més senzilla. No revisar el codi periòdicament seria una gran equivocació perquè, al aparèixer un error, s'hauria de revisar la major part del projecte i ens faria perdre molt temps.

Amb tota la informació que tenim, podem afirmar que utilitzarem el procés de metodologia àgil Scrum per a realitzar el projecte. (15)



Il·lustració 2: Metodologia Scrum

3.4 Eines de desenvolupament i seguiment

El programa de gestió de recursos empresarials[6] que fem servir és el *SAP Business One* (o SBO), aquest gestiona tots els recursos de l'empresa mitjançant una complexa estructura de bases de dades.

Per tal de poder accedir i treballar correctament amb aquesta estructura de bases de dades necessitarem un bon gestor de bases de dades. Utilitzarem l'*SQL Server Managment Studio* (o SSMS).

L'eina principal alhora de desenvolupar el projecte serà el *Visual Studio* de Microsoft. És un entorn que ve amb el conjunt d'eines necessàries per dissenyar, desenvolupar, depurar i testejar el nostre programa. Es fa servir aquest entorn ja que és totalment compatible amb SBO i SSMS i sabem que no ens trobarem amb cap sorpresa al llarg del projecte.

Finalment, és necessari poder fer un bon seguiment del projecte tal que compleixi amb la metodologia esmentada anteriorment. Per aquest projecte utilitzarem el *Team Foundation Server* de Microsoft. Aquest servei, permet emmagatzemar el codi al núvol i compartir-lo gratuïtament entre 5 usuaris. Ens permetrà veure totes les versions del codi penjat i els comentaris que s'hagin guardat amb cadascuna d'elles. També s'hi poden assignar tasques per tal de veure que tenim pendent i que hem finalitzat.

L'ús d'aquest servei paral·lelament amb les reunions amb el director i el ponent del projecte, ens permetrà portar a terme un seguiment adequat del nostre treball. A més a més, també és una eina de Microsoft, així que és totalment compatible amb la resta.

Un cop definida la part de software que utilitzarem, només falta definir el hardware, la màquina que es farà servir per a realitzar el projecte serà un Lenovo ThinkPad T440.

[6] Enterprise Resource Planning o ERP

4 Planificació temporal

Aquest és un projecte amb una durada d'aproximadament 4 o 5 mesos, des de principis de Setembre a finals de Gener. Per tal de tenir-lo acabat en el temps establert cal realitzar una bona planificació. Veiem a continuació les tasques en que es divideix.

4.1 Descripció de les tasques

Al fer servir una metodologia àgil, no s'hauran de seguir les tasques establertes de forma molt estricta. La idea és que cada una o dues setmanes es vagin definint un conjunt d'objectius a assolir per tenir un control de com evoluciona la tasca però que, alhora, permeti fer alguna variació en cas de que sorgeixi algun problema o que s'introdueixi una nova idea.

Tot i que no s'hagin de seguir les tasques de forma estricta, cal tenir en compte que hi haurà algunes tasques les quals seran vitals per poder avançar en el projecte com per exemple, acabar correctament la tasca d'obtenció i definició de dades abans de començar a implementar l'algorisme.

Les tasques a assolir són les següents:

1. Fita inicial
2. Anàlisi i disseny del projecte
3. Configuració de l'entorn
4. Obtenció de dades
5. Implementació de l'algorisme
6. Fita final

Veure diagrama de Gantt a l'Annex.

4.2 Fita inicial

És la part inicial del projecte, a la qual es fa un anàlisi general del treball a realitzar i de la forma en que evolucionarà. Es fa durant els primers mesos i consta de les següents parts:

1. Definició de l'abast i contextualització
2. Planificació temporal
3. Gestió econòmica, social i sostenibilitat
4. Presentació preliminar
5. Plec de condicions
6. Document final
7. Documentació presentació

4.3 Anàlisi i disseny del projecte

A la part de l'anàlisi és important establir bé quins són els objectius a assolir durant el projecte, quins requisits s'han de complir i quines funcionalitats s'han d'implementar.

A la part del disseny s'ha de plantejar be la generació de l'algorisme encarregat d'interpretar el conjunt de clients, vehicles i comandes i retornar un resultat. Quines estructures de dades farem servir, i com estructurarem el codi per tal de que sigui llegible i flexible.

També cal definir la interfície gràfica amb la que executarem aquest algorisme i amb la que interactuarem amb *SAP Business One*. Tot això implica que haurem de fer ús dels coneixements que s'han adquirit al llarg del grau i a la feina.

4.4 Configuració de l'entorn

Una tasca senzilla, on simplement s'han de configurar els programes que farem servir al llarg del projecte que són, el *Visual Basic*, l'*SQL Server* i el *SAP Business One*.

4.5 Obtenció de dades i implementació de l'algorisme

Les tasques més importants del projecte, ja que són les que tenen mes pes. Hem vist com funcionarà l'obtenció de dades del projecte a la secció de l'abast i hem vist quin serà l'algorisme que implementarem a la secció de l'estat de l'art.

Al ser dues tasques tant importants es poden segmentar en 4 fases. Una primera fase d'anàlisi, una segona de disseny, una tercera d'implementació i finalment una última part reservada a les proves.

4.6 Fita final

A la fita final es presentarà la memòria conjuntament amb un apartat d'annexos, i el codi de l'aplicació.

4.7 Duració aproximada

Tasca	Duració (hores)
Fita inicial	**90**
Anàlisi i disseny	**40**
Configuració entorn	**10**
Obtenció de dades	**30**
Generació graf	**200**
Altres funcionalitats	**30**
Fita final	**40**
Total	**440**

Taula 1: Duració aproximada del projecte

4.8 Valoració d'alternatives

Durant la realització del projecte hi ha la possibilitat de que apareguin vàries desviacions respecte la planificació original. Aquestes s'hauran d'analitzar i gestionar per tal que l'impacte en el projecte sigui el menor possible. Gràcies a l'ús d'una metodologia àgil a on es fixen un conjunt de fites a complir de forma periòdica, podrem controlar aquestes desviacions, gestionar-les i corregir-les fàcilment.

Com hem vist anteriorment a la taula de duració aproximada, tenim les hores fixades per a cadascuna de les tasques. Aquestes durades poden variar i fins i tot solapar-se ja que no es poden preveure exactament tots els imprevistos amb els que ens podem trobar. La solució per a realitzar un bon seguiment de l'evolució del projecte i assegurar el correcte compliment de cadascuna de les tasques, seran les reunions que es faran amb el director i ponent del projecte.

Hem vist que gràcies a la metodologia àgil podem controlar fàcilment els errors que sorgeixin complint amb els terminis establerts. Quant als recursos, que hem vist a la secció anterior de l'abast, no haurien de representar cap problema ja que cadascun d'ells disposa d'una llicència d'ús il·limitada.

5 Impacte econòmic i sostenibilitat

5.1 Àrea econòmica

Aquest apartat proporcionarà l'estimació del cost del projecte tenint en compte recursos humans, hardware, software, llicències, despeses indirectes imprevists i contingències.

5.1.1 Pressupost de recursos humans

Hi ha 4 posicions en aquest projecte: director, dissenyador, desenvolupador i tester. Com que el projecte el realitza només una sola persona, aquesta serà l'encarregada de fer totes les tasques. Veiem a la següent taula les tasques i el seu cost amb més detall:

Rol	Hores	Preu / Hora	Preu Total
Director de projecte	100 h	60€/h	6.000€
Dissenyador	50 h	40€/h	2.000€
Desenvolupador	250 h	50€/h	12.500€
Tester	40 h	30€/h	1.200€
TOTAL	**440 h**		**21.700€**

Taula 2: Pressupost de Recursos Humans

5.1.2 Pressupost de Hardware

Per realitzar les tasques d'implementació, proves i documentació ens farà falta només un element de hardware. En podem veure el pressupost a continuació.

Element	Preu	Unitats	Vida Útil	Amortització
Lenovo ThinkPad T440	1.500€	1	5 anys	300€
TOTAL	**1500€**			**300€**

Taula 3: Pressupost de Hardware

5.1.3 Pressupost de software

També seran necessàries les següents eines de software.

Element	Preu	Unitats	Vida Útil	Amortització
Microsoft Office	69€	1	1 any	69€
Team Foundation Server	0€	1	-	0€
SAP Business One SDK[7]	10.000€	1	-	0€
SQL Server Studio	0€	1	-	0€
Visual Studio	3.328€	1	-	0€
TOTAL	**12.276,5€**			**69€**

Taula 4: Pressupost de Software

5.1.4 Pressupost de llicències

Un altre recurs que cal tenir en compte, apart del software i el hardware, són les llicències que cal comprar. Per a SQL Server Studio són gratis, i la del Visual Studio va inclosa amb el preu del producte. Però la llicència de SAP s'ha de renovar cada any. Podem veure el cost a la següent taula:

Element	Preu	Unitats	Vida Útil
SAP Business One	2500€	1	1 any
TOTAL	**2500€**		

Taula 5: Pressupost de llicències

5.1.5 Despeses indirectes

A tots els projectes s'han de tenir en compte les despeses que es generen al fer ús d'elements com l'electricitat o el paper.

Element	Preu	Unitats	Cost estimat
Electricitat	0,08€/kWh	65.000 kWh	5.240€
Paper	20€/pack	1 pack	20€
TOTAL			**5.260€**

Taula 6: Pressupost despeses indirectes

[7] Preu de compra de llicències de programació per varis usuaris.

5.1.6 Pressupost imprevists

A tots els projectes s'ha de tenir en compte que poden sorgir imprevists i per això s'han d'haver destinat uns diners per a solucionar-los.

Imprevist	Preu
S'espatlla l'ordinador	1.500€
TOTAL	**1.500€**

Taula 7: Pressupost imprevists

5.1.7 Pressupost contingències

A tots els projectes s'ha de destinar una partida per a les contingències. Ens podem trobar que si per culpa d'algun problema no es compleix amb el termini d'entrega o alguna cosa no funciona com és degut el comprador prengui accions en contra nostra. En aquest cas destinem diners per errors de programació i/o baixes del programador.

Risc acceptat	% Ocurrència	Cost estimat conseqüències	Exposició al risc
Baixa programador	10%	400€/dia baixa	40€/dia baixa
Errors programació	20%	400€/error	80€/error
TOTAL			120€
TOTAL (10 dies)			**1.200€**

Taula 8: Pressupost contingències

5.1.8 Pressupost final

Ara ja hem vist les diferents parts del pressupost, a la següent taula ho posarem tot junt per veure el pressupost final del projecte.

Concepte	Preu
Recursos humans	21.700€
Hardware	1.500€
Software	12.276,5€
Llicències	2500€
Despeses indirectes	5.260€
Imprevists	1.500€
Contingències	1.200€
TOTAL	**45.936,5€**

Taula 9: Pressupost final

5.2 Àrea social

Aquest projecte treballa conjuntament amb un sistema de gestió de recursos empresarials molt conegut, el *SAP Business One* (o SBO). Un programa que utilitzen moltes empreses a Catalunya, Espanya i a la resta del món.

El projecte automatitzarà una part de la logística d'una empresa, encarregada de servir comandes a un conjunt de clients, gràcies a un programa extern que treballarà conjuntament amb el SBO. Així doncs, qualsevol empresa que gestioni els seus recursos amb el SBO i que tingui la necessitat de millorar aquesta part de la seva gestió de logística hi podria estar interessada.

Al mercat hi ha altres empreses que han desenvolupat aquest programa, però com que hi ha molta competència, és interessant que hi hagi vàries solucions, així les empreses que ho necessiten poden escollir. A més a més, el conjunt de solucions no seran ben bé iguals, cadascuna tindrà algun benefici respecte les altres, tal que, l'empresari que hi estigui interessat, podrà escollir la que millor s'ajusti a les seves necessitats.

5.3 Àrea ambiental

Al ser un projecte de software, a l'àrea ambiental no hi ha gaire res a comentar. L'únic que pot afectar al medi ambient són les despeses indirectes, que podem trobar a l'apartat 5.1.5.

Bàsicament els processos d'obtenció de llum i paper són els que afecten el medi ambient. Però és tant poca la despesa que realitzem en aquest projecte, que no és necessari entrar més en detall.

5.4 Informe sostenibilitat

L'informe de sostenibilitat ens mostra que econòmicament la previsió ha estat bona, els resultats al final del projecte coincideixen amb la previsió inicial. L'impacte social és mínim tal com s'havia previst i l'impacte ambiental ha set menor del que es va preveure al iniciar el projecte, s'han consumit menys recursos.

Sostenible ?	Econòmica	Social	Ambiental
Planificació	Viabilitat Econòmica	Millora en la qualitat de vida	Anàlisis de recursos
valoració	8	4	9
Resultats	Cost final versus previsió	Impacte en entorn social	Consum de recursos
valoració	8	4	7
Riscs	Adaptació a canvis d'escenari	Danys socials	Danys ambientals
Valoració	0	0	0
Valoració total	16	8	16

Taula 10: Informe sostenibilitat

6 Desenvolupament

En aquest apartat veurem més en detall com s'ha realitzat la implementació de l'algorisme *TABU Search* i coneixerem la interfície que s'ha dissenyat per donar-hi accés des de *SAP Business One*.

6.1 Implementació TABU Search

Aquesta metaheurística parteix d'una solució inicial i cerca una millor solució executant dos processos, la intensificació i la diversificació. El primer executa una heurística de cerca local i intenta trobar la millor solució d'un veïnat, el segon s'encarrega de desencallar la solució un cop aquesta s'ha quedat estancada localment. L'algorisme finalitza al complir-se un criteri d'aturada.

6.1.1 Solució inicial

Per obtenir la solució inicial ens hem basat en una heurística d'inserció, el *Push Forward Insertion Heuristic* (PFIH) que hem vist a l'apartat 2.3.1.2. Inicialment tenim un vector de clients amb tots els clients pendents d'assignar i tenim un vector de rutes buit. L'algorisme inicialitza una ruta col·locant el magatzem i un primer client, aquest primer client serà el que tingui la finestra de temps amb el menor horari de tancament per tant, el que volem servir abans.

Un cop inicialitzada la ruta s'agafarà el següent client i es calcularà el cost d'inserir-lo a totes les posicions possibles de la ruta sempre complint les restriccions. De totes les posicions ens quedarem amb la millor bastant-nos en els resultats obtinguts a les funciones comentades a 2.3.1.2. A l'estructura Position guardem la posició on inserim el client a la ruta i el cost d'aquesta inserció calculat amb $ComputeMinimumPosition(\dots)$

```
Position feasibility;
for (unsigned int i = 0; i < toAssignBP.size(); ++i)
{
    // Check capacity constraint and compute if satisfied
    if (EnoughCapacity(routeBP, toAssignBP[i].Capacity()))
    {
        std::pair<unsigned int, double> aux = ComputeMinimumPosition(toAssignBP[i], routeBP, routeBP, -1);
        if (aux.first != -1) feasibility.push_back(std::make_pair(i, aux));
    }
}
```

Il·lustració 3: Codi PFIH, càlcul insercions possibles en una ruta

Aquest procés es repetirà per cada client pendent d'assignar, un cop s'hagin provat tots, agafarem la inserció més òptima bastant-nos amb les fórmules de 2.3.1.2. L'assignarem a la ruta i tornarem a iniciar el procés del paràgraf anterior. Quan aquest procés no troba cap nou client a inserir a la ruta, aquesta es guarda al vector de rutes i se'n genera una de nova. L'algorisme finalitza quan s'han inserit tots els clients.

```
int PFIH::ComputeOptimiumBP(const Position &feasibility, const std::string &cardCode)
{
    std::vector<double> optimum;
    for (unsigned int j = 0; j < feasibility.size(); ++j)
    {
        double dou = dm.find(cardCode + "-" + toAssignBP[feasibility[j].first].CardCode())->second.first;
        dou *= lambda;
        dou -= feasibility[j].second.second;
        optimum.push_back(dou);
    }
    return Maximum(optimum);
}
```

Il·lustració 4: Codi PFIH, càlcul inserció òptima

6.1.2 Intensificació

Un cop definida la solució inicial és moment d'intensificar. La solució inicial són un conjunt de rutes que anomenem veïnat. L'objectiu és intercanviar clients entre aquest veïnat amb l'objectiu de millorar el cost general de la solució. L'algorisme encarregat de fer els intercanvis és el que hem vist a l'apartat 2.3.1.3. En aquest projecte fem servir $\Lambda = 2$ per tant, tenim els següents vuit intercanvis a realitzar $(0,1), (1,0), (1,1), (2,0), (0,2), (2,1), (1,2), (2,2)$.

Per aclarir la definició d'intercanvi veiem dos exemples, cal tenir en compte que els intercanvis sempre són entre dues rutes del veïnat. Si fem un intercanvi $(0,1)$ entre les rutes $(R1, R2)$ vol dir que agafem un client de $R2$ i l'inserim a $R1$. Si fem un intercanvi $(2,1)$ vol dir que agafem dos clients de $R1$ i els inserim a $R2$, i que agafem un client de $R2$ i l'inserim a $R1$. Els dos clients que inserim a una mateix ruta no necessàriament han de ser seqüencials.

L'algorisme s'executa indefinidament fins a explorar tot el veïnat i no trobar cap millora. Una exploració del veïnat consisteix en agafar cadascuna de les rutes i intercanviar tots els seus clients posició a posició, amb cadascun dels intercanvis definits per Λ.

A mesura que anem explorant el veïnat, s'han de guardar els moviments i les solucions que obtenim a la $llista\ tabú$ i també les solucions amb *Global Best* (GB) a la $llista\ de\ candidats$ per a futures intensificacions. L'estructura de la $llista\ tabú$ normalment té una mida petita ja que si és molt gran llavors no és massa útil alhora d'evitar cicles. Tot i així com que normalment és superior a 8 s'ha decidit utilitzar l'estructura $std::map$. S'ha escollit aquesta ja que s'hauran de realitzar moltes consultes tant a la $llista\ tabú$ com a la $llista\ de\ candidats$ i la utilització d'un mapa fa que sigui més àgil que un vector a l'hora de realitzar comparacions.

A la següent il·lustració podem veure com recorrem tot el conjunt de rutes, apliquem els moviments d'intercanvi amb les funcions $Movement10(\dots)$ i $Movement01(\dots)$. Verifiquem que la solució segueixi sent correcta i passem a la següent capa on s'apliquen la resta d'intercanvis. Al final tenim la variable $tabuFreqHit$, que és una altra forma de sortir de la intensificació, indica que s'ha realitzat un mateix moviment o s'ha visitat una mateixa solució masses vegades, és a dir, controla la freqüència.

```
// For a complete search cycle (whole neighbourhood)
for (int i = 0; i < copy.size() - 1; ++i)
{
    for (int j = i + 1; j < copy.size(); ++j)
    {
        // Apply interchanges to 2 routes, using GB method
        if (Movement10(copy[i], copy[j], i, j, false, copy))
        {
            VerifySolution(copy, distance, exit, finalRes);
            SecondLayerMoves(copy, distance, exit, finalRes, i, j);
            copy = routes;
        }

        if (Movement01(copy[i], copy[j], i, j, false, copy))
        {
            VerifySolution(copy, distance, exit, finalRes);
            SecondLayerMovesInverse(copy, distance, exit, finalRes, i, j);
            copy = routes;
        }
    }
    if (tabuFreqHit) break;
}
```

Il·lustració 5: Intensificació, fragment de l'estructura general

6.1.3 Diversificació

Amb la solució inicial ja intensificada, disposem d'un conjunt de rutes que no es pot millorar, aquí és on entra en joc la diversificació. De forma aleatòria es seleccionen dues rutes del conjunt i s'apliquen intercanvis de clients també de forma aleatòria, sempre complint les restriccions, però sense importar si la solució empitjora.

Es fan servir dues funcions, la primera és $MovementFromToRandom(\dots)$, agafa dues rutes aleatòries i dos valors aleatoris que indicaran el tipus de intercanvi a realitzar, és a dir, quin intercanvi dels disponibles segons Λ aplicarem. Els clients a intercanviar entre les rutes i el moviment d'intercanvi es seleccionen de forma aleatòria.

La segona funció es diu $Relink(\dots)$, el funcionament és el mateix que l'anterior però aquí els clients es mouen en bloc, és a dir, si tenim un moviment $(2,2)$, haurem de moure dos clients de cada ruta de forma seqüencial, i els haurem de col·locar a la nova ruta junts. La funció anterior en canvi permetia col·locar-los per separat.

Aquestes funcions es van executant aleatòriament sobre el veïnat i van col·locant nous candidats a intensificar a la llista. El procés de diversificació finalitza quan s'han obtingut un nombre suficient de candidats o quan s'han realitzat un nombre fix d'iteracions.

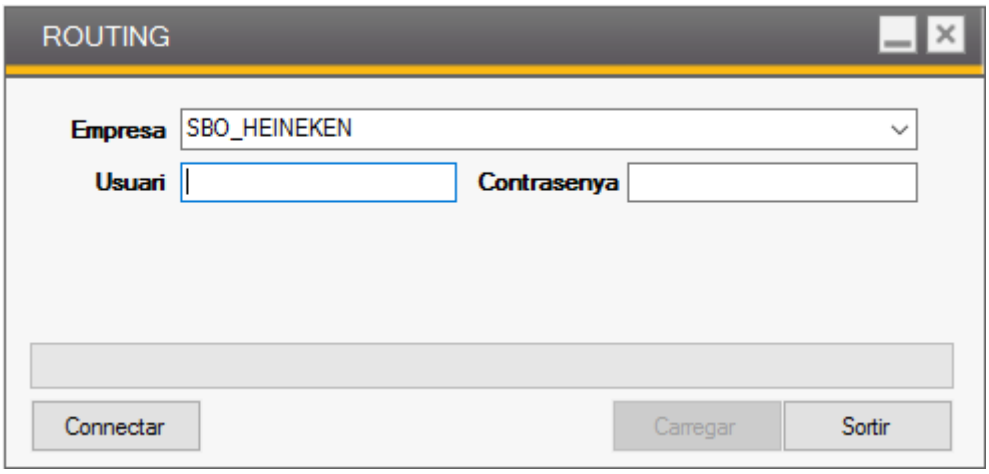
6.1.4 Criteri d'aturada

Acabada la diversificació, seleccionem el millor candidat de la llista i tornem a començar la intensificació. Aquest procés s'ha d'anar repetint fins que es compleixi un criteri d'aturada. En aquest projecte en tenim dos, el primer finalitza l'execució quan s'han realitzat un nombre màxim d'iteracions, el segon i el que utilitzarem, atura quan durant un conjunt de iteracions no s'ha obtingut una millora d'un tant per cent com per exemple, si en 5 iteracions la solució no millora un 3%, aturem l'algorisme.

6.2 Interfície amb SAP Business One

Un cop definit l'algorisme és hora de connectar amb *SAP Business One* (o SBO) i extreure la informació de vehicles, clients, distàncies i comandes. Per a fer-ho generem una interfície amb .NET que ens connecta a la BBDD corresponent i genera un fitxer de text amb la informació necessària per enviar a l'algorisme.

A la imatge següent podem veure la interfície que ens demana l'empresa a on ens volem connectar, l'usuari i la contrasenya. Un cop establerta la connexió, s'habilita el botó Carregar que s'encarrega de llegir els vehicles disponibles, els clients amb comandes pendents de servir i calcular les distàncies entre cadascun dels clients.



Il·lustració 6: Interfície amb SAP Business One

Un cop carregada la informació, és moment d'executar l'algorisme. Per executar-lo des de la interfície s'ha generat un llibreria que cridem amb les 4 funcions que veurem a continuació. D'aquesta forma podem enviar la informació necessària llegida a SBO a la llibreria, executar l'algorisme i obtenir els resultats.

A la primera imatge veiem com s'han definit les funcions al codi C++ per tal de poder ser cridades des de l'aplicació .NET i a la següent veiem la crida des de l'aplicació.

```
/**
 * @brief Creates a Computing class instance from VB.NET
 */
void * __stdcall CreateComputing()
{
    return new Computing;
}

/**
 * @brief Destroys the Computing class instance from VB.NET
 */
void __stdcall DestroyComputing(void* objptr)
{
    Computing * computing = (Computing *) objptr;
    if (computing) delete computing;
}

/**
 * @brief Sends parameters from VB.NET
 *
 * @param params Parameters from VB.NET separated by commas
 */
void __stdcall SendParametersComputing(void* objptr, LPSTR params)
{
    Computing * computing = (Computing *) objptr;
    if (computing) computing->SendParameters(params);
}

/**
 * @brief Iniciates algorithm execution from VB.NET
 */
void __stdcall ExecuteAlgorithmComputing(void* objptr)
{
    Computing * computing = (Computing *) objptr;
    if (computing) computing->ExecuteAlgorithm();
}
```

Il·lustració 7: Funcions llibreria C++

```
Private Declare Function CreateComputing Lib "Computing.dll" () As UInteger
Private Declare Sub DestroyComputing Lib "Computing.dll" (ByVal objptr As UInteger)
Private Declare Sub SendParametersComputing Lib "Computing.dll" (ByVal objptr As UInteger, <MarshalAs(UnmanagedType.LPStr)> ByVal fp As StringBuilder)
Private Declare Sub ExecuteAlgorithmComputing Lib "Computing.dll" (ByVal objptr As UInteger)
```

Il·lustració 8: Funcions llibreria .NET

7 Resultats

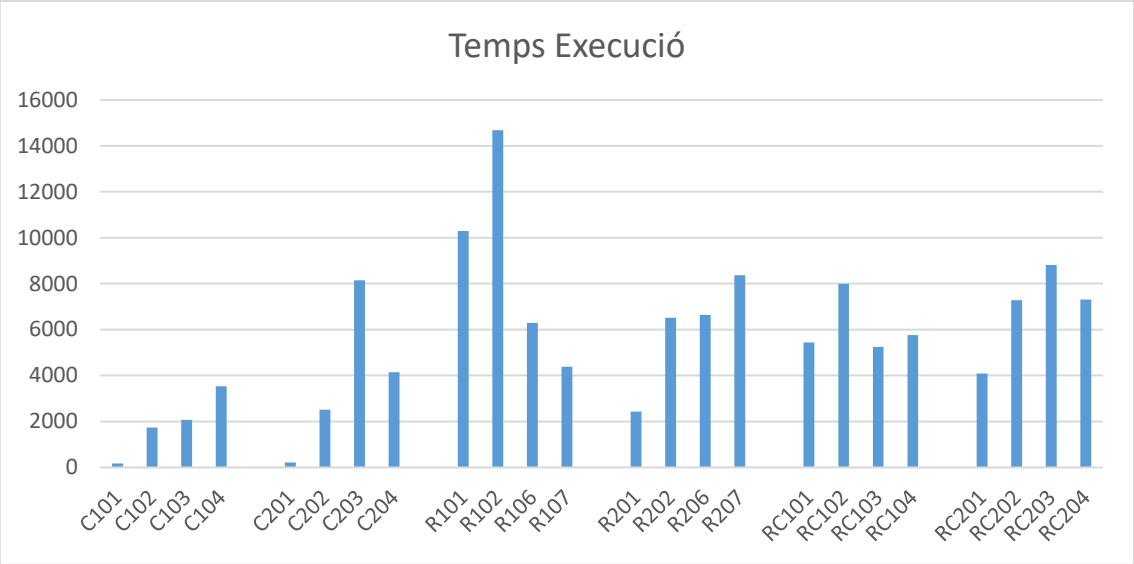
En aquesta secció analitzarem el funcionament del nostre algorisme solucionant un conjunt de problemes ficticis o benchmarks plantejats per Marius Solomon (16). Aquest conjunt de benchmarks són uns estàndards que històricament s'han fet servir per avaluar algorismes que intenten solucionar el problema *TABU Search* amb el que es basa aquest projecte.

Totes les proves s'han realitzat amb un Lenovo T440 de 8GB de RAM. Els resultats d'aquestes proves ens donaran una idea del funcionament de l'algorisme ja que, podrem contrastar els nostres resultats amb els millors resultats obtinguts fins al moment sobre cadascun dels problemes. Addicionalment també podrem comparar els nostres resultats amb l'execució d'un algorisme *TABU Search*, anomenat *S-Tabu* (12).

7.1 Resultats comparativa

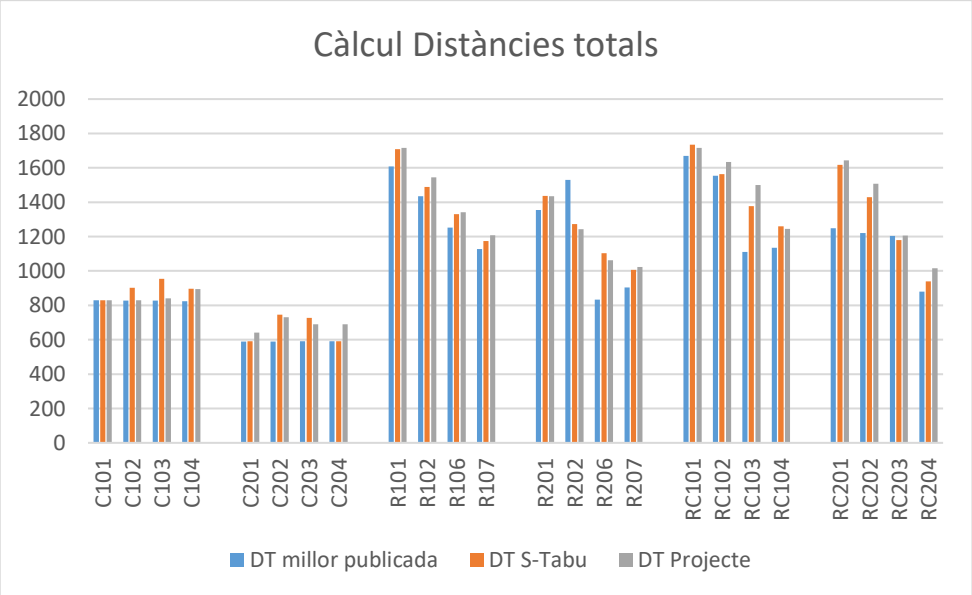
Hem provat un conjunt de 24 problemes que es divideixen en 6 categories, $C1, C2, R1, R2, RC1\ i\ RC2$. Els problemes de la categoria C són problemes amb les dades en clúster, és a dir, els nodes estan organitzats en clústers geogràfics o de finestres de temps. Els de la categoria R representen dades distribuïdes uniformement i els de la categoria RC són un híbrid entre R i C. La diferència entre 1 i 2 són variacions en les finestres de temps.

L'execució del programa per a cadascun del problemes és bastant ràpida, podem comprovar els resultats a la taula de la següent pàgina, els temps són en milisegons, no hi ha cap execució que superi el minut ni que s'hi aproximi.

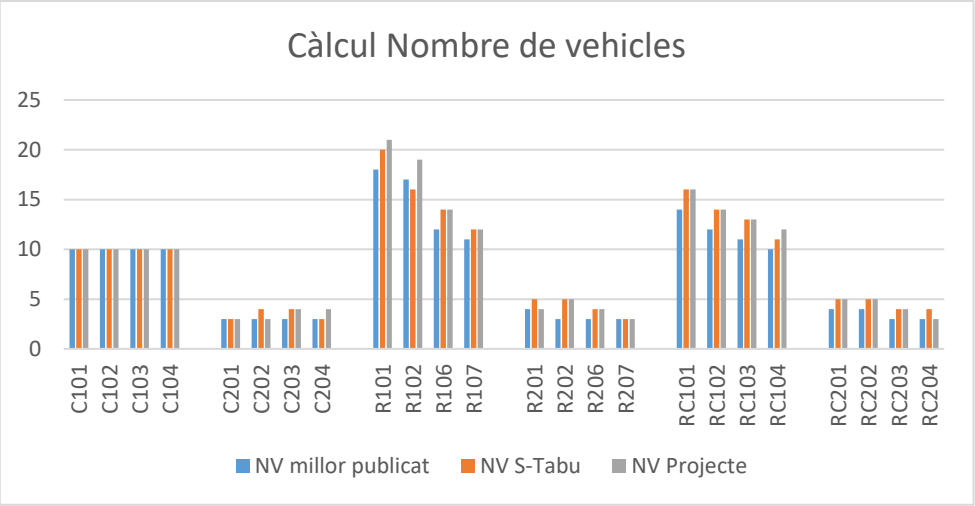


Il·lustració 9: Temps d'execució en ms.

Disposem de la millor distància total publicada i el nombre mínim de rutes millor publicades per cadascun dels problemes. També disposem dels resultats d'un article on han utilitzat el mètode de *TABU Search*, ells l'anomenen *Strict TABU* (o S-TABU) (12), i finalment tenim els nostres resultats. Veurem dues gràfiques, la de distància total (DT) i la del nombre de vehicles (NV).



Il·lustració 10: Càlcul de distàncies totals per cadascun dels problemes.

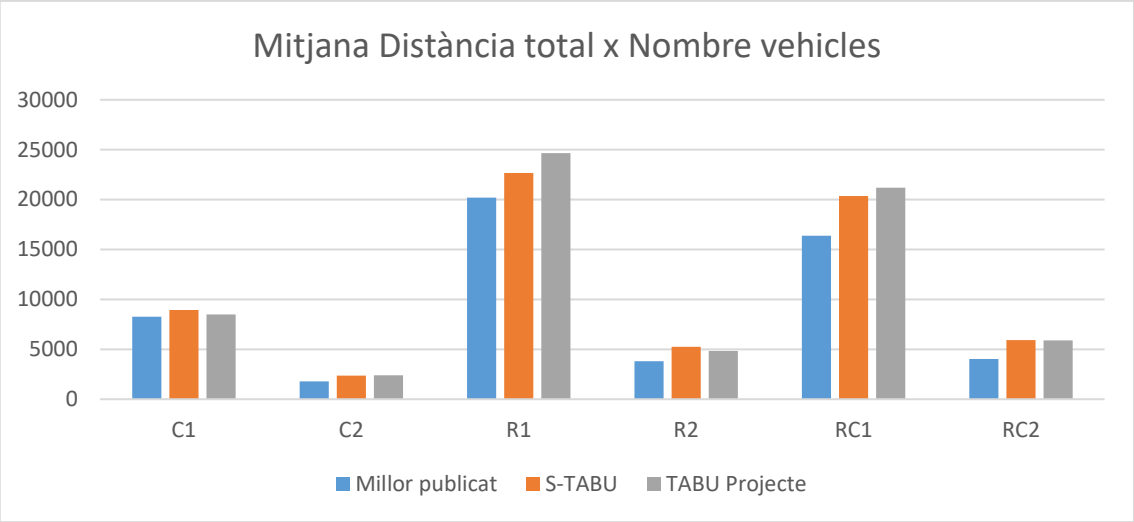


Il·lustració 11: Càlcul de nombre de vehicles totals.

Com podem veure a les gràfiques anteriors, els resultats de l'algorisme del nostre projecte són bastant correctes en comparació a la resta, tan per nombre de vehicles com per distància. Els resultats en comparació amb l'altre algorisme de *TABU Search* són bastant semblants, en alguns casos fins i tot s'aconsegueixen millors resultats.

Per comentar algun cas veiem per exemple, el problema $R202$, on podem veure que les distàncies totals obtingudes pels algorismes de *TABU Search* són millors, però és degut a que el nombre de vehicles millor publicat és bastant més baix. Si comprovem l'execució $C101$ podem veure que les execucions que utilitzen *TABU Search* són capaces d'obtenir un resultat igual al millor publicat.

Finalment per tal de facilitar la interpretació dels resultats obtinguts, veiem una gràfica de la mitjana de la distància total multiplicada pel nombre de vehicles per cadascuna de les categories. Podem veure com la nostra execució supera l'altra en alguns casos i no s'allunya gaire dels millors resultats publicats.



*Il·lustració 12: Càlcul del nombre de vehicles * distància total.*

8 Conclusions

Amb el treball ja finalitzat és moment de veure si s'ha complert la planificació temporal de la secció 4. La planificació consta de: una fita inicial, que és l'assignatura de GEP, l'anàlisi i disseny del projecte, la configuració de l'entorn, l'obtenció de dades, la implementació de l'algorisme i una fita final on es redacta aquesta memòria.

Considero que la planificació s'ha complert adequadament, l'anàlisi del projecte, la obtenció de dades i la programació de l'algorisme han requerit realitzar un bon aprenentatge sobre les heurístiques i les metaheurístiques, conceptes que havíem vist al llarg de la carrera però molt vagament, gràcies a aquest projecte he obtingut un bon coneixement sobre el tema.

Addicionalment, el projecte m'ha ajudat a aprofundir en els meus coneixements amb el llenguatge C++ i l'entorn de Visual Studio. Per tal de que el programa fos eficient en quant a temps d'execució, apart de realitzar una bona implementació de l'algorisme, he hagut d'aprendre a configurar el compilador de Visual Studio.

Personalment crec que he après molt amb aquest projecte. El fet de poder veure l'evolució del projecte, la implementació dels diferents algorismes, que n'acaben formant un de sencer i finalment, la connexió amb el *SAP Business One* ha estat una experiència molt gratificant.

9 Treball futur

El projecte realitzat té una durada aproximada d'uns 4 o 5 mesos, això fa que tinguem poc temps per desenvolupar l'algorisme, ja que una bona part d'aquest temps l'ocupen la fita inicial, l'anàlisi i l'obtenció de dades. Així doncs no s'han pogut afegir al programa totes les funcionalitats necessàries, en aquest apartat veurem de quina manera podem millorar el programa:

- Afegir una nova restricció a cadascun dels vehicles apart del pes, el volum. Això implicaria agafar les dimensions del articles a servir i també de tots dels camions. A partir d'aquí s'hauria de realitzar un càlcul de volums per decidir quins articles es poden carregar i quins no independentment de que hi hagi capacitat suficient.

- Permetre canviar els paràmetres de configuració de l'algorisme *TABU Search* com la mida de la $llista\ tab$ú i altres des de la interfície. Així cada usuari podrà escollir el tipus de funcionament que millor s'ajusti a les seves necessitats.

- Millorar la obtenció de dades de *Google Maps*, ara simplement es calcula la distància d'un client a un altre, podríem afegir-hi informació sobre el trànsit i veure quines hores són les més dolentes per servir a certes zones i afegir-ho com a restricció, sempre complint amb les finestres de temps de cadascun dels clients.

10 Bibliografia

1. **Matai, Rajesh, Singh, Surya i Lal mittal, Murari.** *Traveling salesman Problem: an Overview of Applications, Formulation, and Solution Approaches.* s.l. : http://www.intechopen.com/books/traveling-salesman-problem-theory-and-applications/traveling-salesman-problem-an-overview-of-applications-formulations-and-solution-approaches, 2010.

2. **Wikipedia.** Travelling salesman problem. [En línia] https://en.wikipedia.org/wiki/Travelling_salesman_problem.

3. **SAP.** SAP Business One Help. [En línia] https://help.sap.com/businessone.

4. **Wikipedia.** SAP SE. [En línia] https://es.wikipedia.org/wiki/SAP_SE.

5. **SL, Nexus Geografics.** Routing Reparto. [En línia] http://www.routingreparto.com/.

6. **Solutions, SGI.** SGI Transport. [En línia] http://www.sgisolutions.es/sites/sgitransport/index.asp.

7. **Nanda Kumar, Suresh i Panneerselvam, Ramasamy.** [En línia] 2012. http://file.scirp.org/pdf/IIM20120300003_68227741.pdf.

8. **Wikipedia.** Vehicle Routing Problem. [En línia] https://en.wikipedia.org/wiki/Vehicle_routing_problem.

9. **Ropke, Stefan.** Heuristic and exact algorithms for vehicle routing problems . 2005.

10. **Clarke, G. i Wright, W.** Shceduling of vehicles from a central depot to a number of delivery points. *Operations Research 12.* 1964.

11. **M. Solomon, Marius.** Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research, 35, No. 2.* 1987.

12. **Tan, K.C., et al.** Heuristic methods for vehicle routing problem with time windows. 2000.

13. **Osman, Ibrahim Hassan.** Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem.

14. **Piad, Alejandro.** Stackoverflow. [En línia] http://stackoverflow.com/questions/10445700/what-is-the-difference-between-heuristics-and-metaheuristics.

15. **Gutiérrez, Catalina.** Intelligence to Business. [En línia] 2013. http://www.i2btech.com/blog-i2b/tech-deployment/para-que-sirve-el-scrum-en-la-metogologia-agil/.

16. **M. Solomon, Marius.** VRTPW Benchmark Problems. [En línia] http://web.cba.neu.edu/~msolomon/problems.htm.

11 Annexes

11.1 Diagrama de Gantt

Task Name	Duration	Start	Finish
Fita Inicial	30d	14/09/16	25/10/16
Definició de l'abast i contextualització	6d	14/09/16	21/09/16
Planificació temporal	3d	22/09/16	26/09/16
Gestió econòmica i sostenibilitat	5d	26/09/16	30/09/16
Presentació preliminar	3d	07/10/16	11/10/16
Plec de condicions	10d	12/10/16	25/10/16
Document final	25d	14/09/16	18/10/16
Presentació	3d	15/10/16	18/10/16
Anàlisi i disseny del projecte	12d	15/10/16	31/10/16
Anàlisi	8d	15/10/16	25/10/16
Disseny	5d	25/10/16	31/10/16
Configuració de l'entorn	2d	01/11/16	02/11/16
Visual Basic	2d	01/11/16	02/11/16
SQL Server	2d	01/11/16	02/11/16
SAP Business One	2d	01/11/16	02/11/16
Obtenció de dades	7d	03/11/16	11/11/16
Anàlisi	1d	03/11/16	03/11/16
Disseny	1d	03/11/16	03/11/16
Implementació	4d	04/11/16	09/11/16
Proves	2d	10/11/16	11/11/16
Implementació algorisme	21d	12/11/16	09/12/16
Anàlisi	6d	12/11/16	18/11/16
Disseny	5d	20/11/16	24/11/16
Implementació	5d	25/11/16	01/12/16
Proves	6d	02/12/16	09/12/16
Fita final	36d	10/12/16	27/01/17
Finalitzar projecte	16d	10/12/16	30/12/16
Redactar memòria	23d	10/12/16	10/01/17
Redactar Presentació	9d	10/01/17	20/01/17
Presentació	5d	23/01/17	27/01/17